山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900150221 | 姓名： 张进华 | | 班级： 智能19 |
| 实验题目：数组描述线性表 | | | |
| 实验学时：2 | | 实验日期： 9.28 | |
| 实验目的：   1. 掌握线性表结构、数组描述方法（顺序存储结构）、数组描述线性表的实现。 2. 掌握线性表应用 | | | |
| 软件开发工具：  visual studio code  g++ | | | |
| 1. 实验内容   通讯录中每一个联系人的内容有：姓名、电话号码、班级、宿舍。由标准输入读入联系人信息，使用线性表中操作实现通讯录管理功能，包括：插入、删除、编辑、查找（按姓名查找）；键盘输入一班级，输出通讯录中该班级中所有人的信息。  每个操作的第一个数为操作数(插入-0，删除-1，编辑-2，查找-3，输出一个班所有人员信息-4)，具体格式如下:   * 0 姓名 电话 班级 宿舍 插入一条记录 * 1 姓名 根据姓名删除一条记录 * 2 姓名 编辑项目 项目新值 根据姓名编辑一条记录(编辑项目为1到3的整数，1代表编辑电话，2代表编辑班级，3代表编辑宿舍) * 3 姓名 根据姓名查找，找到输出1，未找到输出0 * 4 班级 输出该班级的所有成员的宿舍号的异或值   其中查找操作当找到相应的人时输出1，未找到输出0。输出一个班级的人员信息时输出所有成员的宿舍号的异或值。输入数据保证合法。   1. 数据结构与算法描述（整体思路描述，所需要的数据结构与算法）   首先定义学生，使用结构体student    作为数组中的每个元素，然后定义线性表类，包含所要实现的四种方法，以及数组容量、元素个数。此处将编辑号码和编辑宿舍号、班级分开，因为实现时前者为int，后者为string。    同时在数组容量过小时采用扩容函数：    最后结果保存在result中输出   1. 测试结果（测试输入，测试输出）      1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   首先插入三条记录，然后按姓名查找，结果显示正确，最后按名字修改对应信息   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）}   #include<bits/stdc++.h>  using namespace std;  struct student //学生类  {      string name;   // 姓名      string phone;  //电话      int class\_num; //班级      int dorm;      //宿舍  };  template <class T>  void changeLength(T \*&arr, int oldLength, int newLength)//改变数组长度  {      T \*temp = new T[newLength];      int number = min(oldLength, newLength);      copy(arr, arr + number, temp); //参数分别是首指针，末指针，目标位置指针      delete[] arr;      arr = temp;  }  class linearlist  {  public:      //构造函数      linearlist(int n = 10);      //析构函数      ~linearlist() { delete[] stu; };      //四种操作      void insert(string &name, string &phone, int &class\_num, int &dorm);      void erase(string &name);      void edit\_phone(string &name, string &new\_info);      void edit\_class\_dorm(string &name, int decision, int &new\_info);      bool serch(string &name);      //计算宿舍号的异或值      int calculate\_xor(int &class\_num);  private:      struct student \*stu;      int len;  //数组容量      int size; //元素个数  };  //构造函数  linearlist::linearlist(int n)  {      stu = new student[n];      len = n;      size = 0;  }  //插入函数  void linearlist::insert(string &name, string &phone, int &class\_num, int &dorm)  {      if (len == size)      {          changeLength(stu, len, len \* 2);          len \*= 2;      }      stu[size].name = name;      stu[size].phone = phone;      stu[size].class\_num = class\_num;      stu[size].dorm = dorm;      size++;  }  //删除函数  void linearlist::erase(string &name)  {      int index = 0; //name对应的元素的索引      while (stu[index].name != name && index < size)          index++;      copy(stu + index + 1, stu + size, stu + index);      stu[--size].~student(); //调用析构函数  }  //编辑号码  void linearlist::edit\_phone(string &name, string &new\_info)  {      int index = 0;      while (stu[index].name != name && index < size)          index++;      stu[index].phone = new\_info;  }  //编辑宿舍班级  void linearlist::edit\_class\_dorm(string &name, int decision, int &new\_info)  {      int index = 0;      while (stu[index].name != name && index < size)          index++;      if (decision == 2)          stu[index].class\_num = new\_info;      else          stu[index].dorm = new\_info;  }  //查找  bool linearlist::serch(string &name)  {      int index = 0;      while (stu[index].name != name && index < size)          index++;      if (index >= size)          return 0;      else          return 1;  }  //宿舍异或和  int linearlist::calculate\_xor(int &class\_num)  {      int result\_xor = 0;      for (int i = 0; i < size; i++)      {          if (stu[i].class\_num == class\_num)              result\_xor ^= stu[i].dorm;      }      return result\_xor;  }  int main()  {      int judge, n, index\_result = 0, i = 0;      string name, phone, new\_phone;      int decision, new\_info, dorm, class\_num;      cin >> n;      int \*result = new int[n];//记录结果      linearlist s;      for (i = 0; i < n; i++)      {          cin >> judge;//输入选项          switch (judge)          {          case 0://0 姓名 电话 班级 宿舍 插入一条记录              cin >> name >> phone >> class\_num >> dorm;              s.insert(name, phone, class\_num, dorm);              break;          case 1://1 姓名 根据姓名删除一条记录              cin >> name;              s.erase(name);              break;          case 2://2 姓名 编辑项目 项目新值              cin >> name >> decision;              if (decision == 1)              {                  cin >> new\_phone;                  s.edit\_phone(name, new\_phone);              }              else              {                  cin >> new\_info;                  s.edit\_class\_dorm(name, decision, new\_info);              }              break;          case 3://3 姓名 根据姓名查找，找到输出1，未找到输出0              cin >> name;              result[index\_result] = s.serch(name);              index\_result++;              break;          case 4://4 班级 输出该班级的所有成员的宿舍号的异或值              cin >> class\_num;              result[index\_result] = s.calculate\_xor(class\_num);              index\_result++;              break;          default:              break;          }      }      for (int j = 0; j < index\_result; j++)      {          cout << result[j] << endl;      }      return 0;  } | | | |